Ingeniería Inversa del Software – Parte I – Herramientas básicas

Por fin ha llegado el tan ansiado primer número de DIFISEC, la revista sobre seguridad informática que escribimos los alumnos de la Facultad de Informática de SS y con él ha llegado también el primer número de esta sección que pretende llamar la atención de los informáticos sobre la importancia de la seguridad vista desde el lado de la inseguridad.

Los primeros artículos que escribiré tratarán sobre la Ingeniería Inversa del Software (IIS) que no es otra que la temida y desconocida disciplina tabú que trata sobre la alteración del comportamiento de los programas compilados modificando su código fuente o variables manejadas.

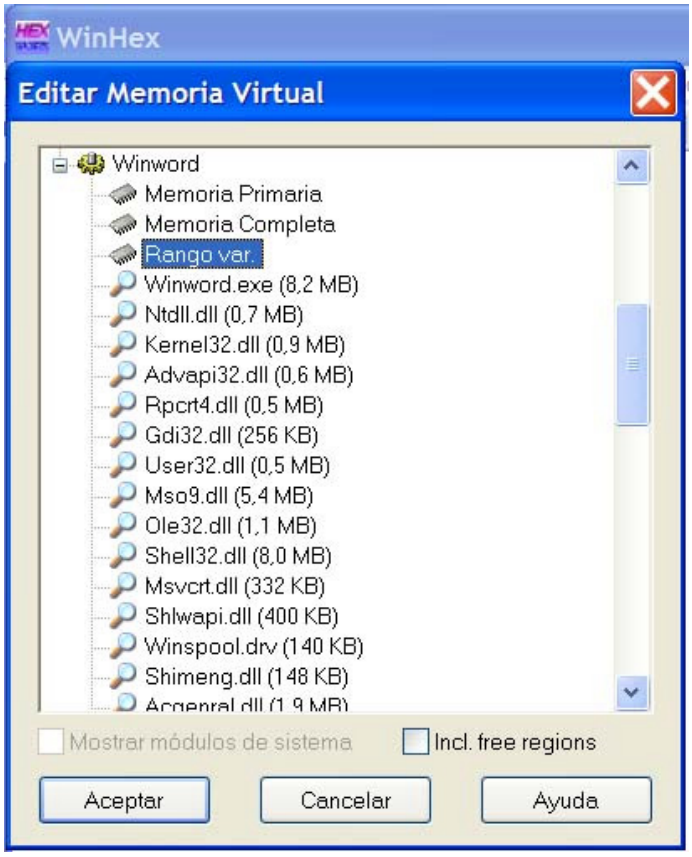
Seguro que más de uno de mis lectores se ha preocupado más de una vez por la seguridad de su código frente a los crackers; y, posiblemente también, la mayoría de ellos ha llegado a la conclusión de que no merece la pena preocuparse porque no pueden concebir ninguna idea capaz de luchar contra la IIS. Es normal que todos nos sintamos inútiles y maniatados frente a temas que nos son desconocidos. Para dar una luz sobre esta oscura disciplina, pretendo explicar en una serie de artículos las ideas en que está basada. Todas ellas son muy sencillas, tal y como veremos en los próximos meses. En cuanto esté publicada esta primera serie de escritos, todos nosotros seremos capaces de modificar el comportamiento de cualquier programa y de esta manera podremos crear código más seguro frente a otra gente con conocimientos similares a los nuestros.

Para empezar a realizar cualquier trabajo, siempre es necesario saber que herramientas se necesitan y qué se puede hacer con ellas. Nuestro caso no es una excepción, por supuesto, y por ello esta primera concatenación de palabras intentará tratar este asunto con la profundidad que se merece.

Una utilidad bastante usada por los crackers más expertos es el editor de memoria. Este tipo de programas permiten localizar y modificar el valor de variables que se encuentran en la memoria RAM del sistema. Como todos sabemos, en estas variables se almacenan todos los datos que maneja un programa, incluso la información sobre si un programa está registrado o cuantos días de prueba quedan.

Aquí no se va a entrar en detalles sobre qué hacer o cómo emplear las herramientas descritas, ya que extendería demasiado el artículo restándole claridad. Eso se tratará más adelante en sucesivos números de esta sección.

El editor de memoria que yo suelo emplear es el WinHex, el cual se puede configurar para que funcione en castellano. En la siguiente figura se puede observar como permite seleccionar que parte de la memoria se va a analizar.



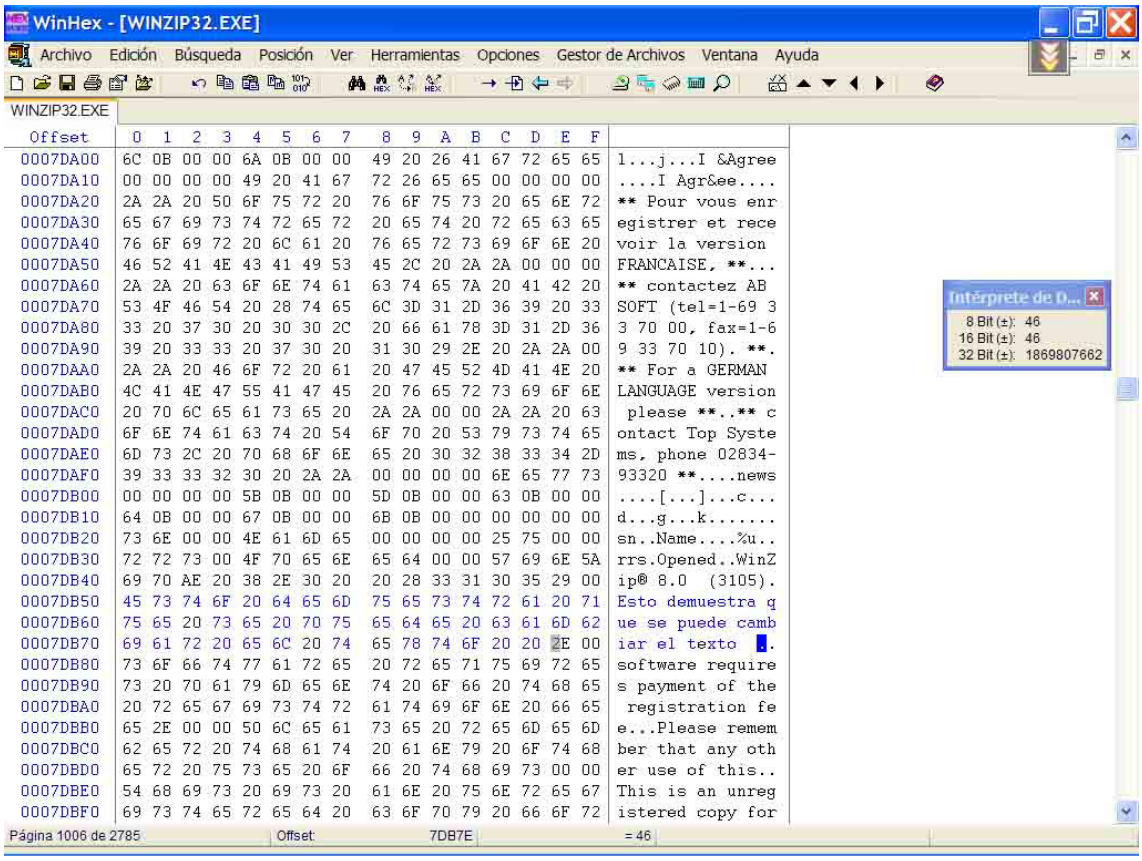
Otra de las utilidades básicas en la IIS es el editor hexadecimal, ese gran desconocido del que todos hemos salido corriendo en cuanto lo hemos visto. Por supuesto, usar un editor hexadecimal sin saber que tienes que cambiar puede ser tan desesperante como tener que hacer cola para no esperar. En fin, como ya veremos, nosotros siempre que lo usemos sabremos de antemano que valores modificar y con qué nuevas secuencias de códigos hexadecimales sustituirlos.

El empleo más típico de este tipo de software es “retocar” de manera perdurable las instrucciones en lenguaje máquina que emplean los programas que ya han sido compilados. Sin ello, sería necesario crackear los códigos de cada programa cada vez

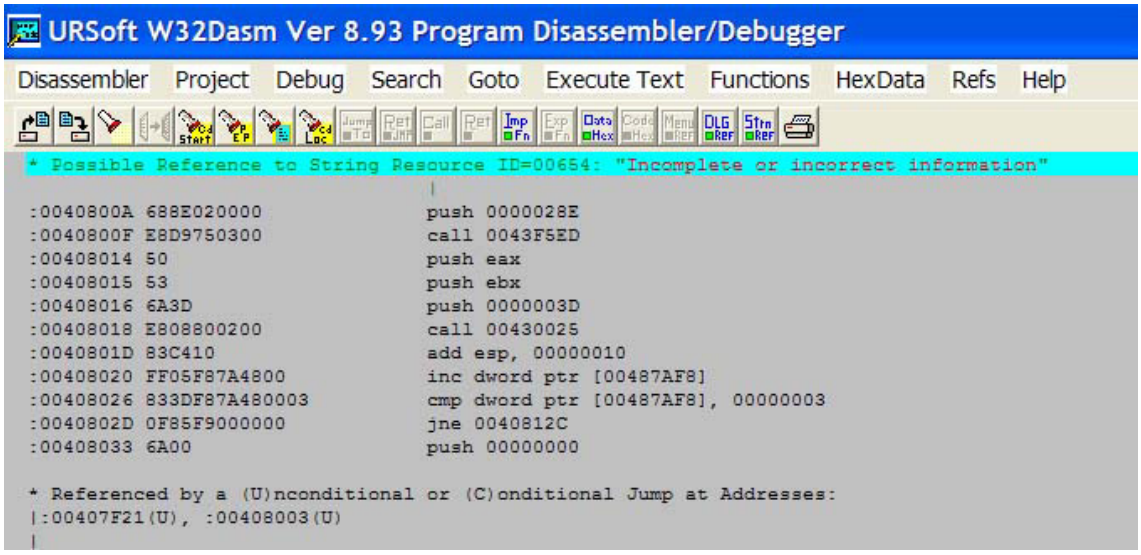
que se quisieran volver a emplear, pues, como ya sabemos, los datos y códigos almacenados en memoria son volátiles y se pierden al cerrar el proceso que los emplea.

El editor hexadecimal que yo suelo emplear es también el WinHex ya que tiene la posibilidad de modificar tanto la memoria como los archivos físicos. ¡Que le voy a hacer si siempre he sido partidario de usar la menor cantidad posible de programas diferentes para realizar la mayor cantidad posible de tareas!

En la siguiente imagen se puede observar el aspecto que presenta cualquier editor hexadecimal:



Para acabar con las herramientas software más importantes y básicas de la IIS mencionaré también el depurador o debugger, otro de los grandes incomprendidos por los no-iniciados. Próximamente dedicaré todo un articulo a este arma tan temido tanto por los desarrolladores como por los crackeadores. Por lo tanto no entraré ahora a detallar sus tipos ni funcionamiento, solo mencionaré que sirve para sacar el código ensamblador de cualquier programa y modificar su comportamiento de manera temporal, al menos en principio. Mi preferido es el W32Dasm porque es el más potente de usar, aunque los hay más sencillos de usar. A continuación se muestra su aspecto:



Esta utilidad software es la más imprescindible y compleja de todas las empleadas en la IIS, por lo que siempre se intentará evitar su uso. Esto es posible gracias a la herramienta más importante de todas las que ha podido desarrollar el hombre a lo largo de toda la historia de la humanidad y, aunque parezca mentira, no me refiero ni a la boina ni a las discotecas, sino a la lógica, el raciocinio o el entendimiento. Esta ha sido la herramienta que ha permitido al ser humano llegar a desarrollarse más que cualquier otro ser viviente del planeta Tierra y, además, ha sido la principal impulsora de todas las revoluciones tecnológicas de la historia. Resulta realmente lamentable que la sociedad actual ya no sea capaz de valorarla como se merece. Aunque parece que todo está ya inventado y es perfecto, esto no es así. En realidad nuestra tecnología no es más que un cúmulo de parches y apaños mal hechos. ¿Cómo se puede explicar que un sistema operativo falle cada dos por tres?¿Cómo es posible que tantas naves espaciales exploten a escasos segundos del despegue?¿A quién le parece lógico que un dispositivo de corte eléctrico como es un diodo, imprescindible para controlar que la corriente eléctrica circule en un solo sentido, cuando se encuentra con una diferencia de potencial inversa de valor mayor al que puede soportar, en ver de estropearse abriendo su circuito, lo haga cortocircuitándose para dejar circular toda esa corriente, que antes retenía, a su libre albedrío? La base tecnológica que tenemos es realmente desastrosa y está en nuestras manos cambiarla, pero para eso necesitamos usar el raciocinio en vez de someter nuestras mentes a la inteligencia o la sabiduría, que solo son importantes para tener una referencia de los fallos existentes y poder evitarlos. Gran parte de los conocimientos existentes en la tecnología actual son vagos, opacos y muchas veces están basados en meras paralogías que somos incapaces de identificar porque nos las

enseñan como axiomas irrefutables, a la vez que indemostrables. Lo único exacto que incluyen son sus bases matemáticas. Posiblemente algún día se superará de nuevo esta época de mentes planas y conoceremos una tecnología más eficaz, a la vez que eficiente. Los ciclos de la historia están de mi lado y estoy seguro de la mayoría de nosotros lo veremos. Es nuestra labor contribuir a ello acostumbrándonos a usar la cabeza para algo más que almacenar conocimientos, en el mejor de los casos.

En el próximo número hablaré de cómo nos puede ayudar la lógica en el tema que estamos tratando. Id abriendo vuestras mentes hasta entonces.

Un saludo a todos y que tengáis un bonito día,

Joxean Nieto

Ingeniería Inversa del Software

- Parte II -

La lógica como herramienta de ataque al software

Como ya comentaba en el primero de los artículos que componen esta colección sobre la ingeniería inversa del software (IIS para abreviar, pero no confundir con Internet Information Services), la lógica puede ser nuestra mejor aliada o nuestra más acérrima enemiga según quien la utilice y según sepa emplearla.

Hoy trataré de abrir un camino a la comprensión sobre como emplearía la lógica un cracker inteligente. De esta manera pretendo sembrar una serie de ideas que servirán para prevenirla más eficazmente. Su aspecto más amigable para nosotros, los programadores, no será tratado en profundidad hasta los últimos artículos, pero en todos ellos se irán desgranando ideas al respecto. Seamos pues sagaces, intentando cazarlas al vuelo según aparezcan, para que las captemos como conclusiones, mucho más perdurables en la memoria que las explicaciones sueltas que daré dentro de unos meses.

Antes de comenzar, aclararé que lo que se explicará aquí solo es un complemento a la IIS, sin formar parte de ella. Esta es una forma rápida, limpia e inteligente de atacar al software evitando aplicar las, a veces tediosas, técnicas de la IIS.

Estos otros métodos de ataque basados en la lógica y el sentido común, aunque solo se pueden aplicar de manera selectiva y en determinados casos, permiten tirar abajo las defensas de un programa sin tener que "pelear" con su código ensamblador. Para ello, simplemente se ataca al recurso de la máquina que sirve como base a la protección del software. De esta manera se evita agredir directamente al programa en sí, con todo lo que ello conlleva. A continuación se enumeran algunos de los tipos de software más fácilmente crackeables de esta manera, a la vez que se explica cual es la base de su defensa y cómo se puede llevar a cabo el ataque:

a) Trials de x dias: son aquellos programas que permiten ser usados en modo de prueba durante un periodo limitado de tiempo. La base de la

máquina que atacará un cracker ante este tipo de programa será el calendario del sistema. Basta un lanzador muy sencillito para hacer que un programa de este tipo funcione para siempre. En el siguiente cuadro se muestra y explica el código en VB .NET de este lanzador:

```
Dim hoy As Date 'VARIABLE GLOBAL DONDE ALMACENARÉ LA FECHA ACTUAL

'ESTA FUNCIÓN SE EJECUTA CON EL EVENTO DE INICIAR EL LANZADOR
Private Sub Form_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    hoy = Today() 'ALMACENO LA FECHA ACTUAL
    Today() = DateValue("23/11/2002") 'PONGO LA FECHA DEL DIA DE LA INSTALACION DEL PROGRAMA
    reloj.Interval = 32000 'ESTABLEZCO QUE SE RECUPERE LA FECHA TRAS 32 SEGUNDOS
    reloj.Enabled = True  'ACTIVO EL TIMER Y EMPIEZA A CONTAR EL TIEMPO
    Shell("C:\Archivos de programa\carpeta\programa.exe") 'LANZO EL PROGRAMA
End Sub

'PASADOS 32 SEGUNDOS SE DA EL EVENTO DEL TIMER CADUCADO:
Private Sub reloj_Elapsed(ByVal sender As System.Object, ByVal e As System.Timers.ElapsedEventArgs) Handles reloj.Elapsed
    Today() = hoy 'RECUPERO LA FECHA ACTUAL
    Me.Close() 'CIERRO EL LANZADOR
End Sub
```

b) Trials de x usos: este sistema de protección permite usar el programa a modo de prueba un número máximo de veces. El recurso a atacar es el contador de usos. Éste se halla, habitualmente, en un fichero de la carpeta del programa o en el registro de Windows. Como dato para encontrarlo se sabe el momento exacto en que se ha modificado por ultima vez. ¡¡¡Puede ser imposible de encontrar si el programador se ha molestado en esconderlo!!!

c) Programas con clave aleatoria: son aquellos que proporcionan un libro de claves o algún otro sistema caro de reproducir. Se le pide al usuario que introduzca una determinada palabra o clave diferente cada vez. Como todas las máquinas emplean el reloj como semilla de aleatoriedad, si se

diseña un lanzador similar al de los trials de x días pero alterando también la hora, se conseguirá que siempre pida la misma clave. Hay que tener en cuenta que la clave cambiará si hay otros programas que no sean los del sistema ejecutándose en segundo plano o si se cambia la velocidad de la máquina.

d) Sistemas de protección patentados: son aquellos que han sido desarrollados con ánimo de lucro, para venderlos a desarrolladores de software que quieran proteger sus propiedades sin perder el tiempo en diseñar una defensa propia. Como están patentados, su funcionalidad esta ampliamente descrita en el documento de la patente y es de dominio público, por lo que es fácil saber a que se debe atacar y como diseñar el crack. La pagina oficial de patentes de España se halla en la siguiente dirección: http://www.oepm.es/. Normalmente, cada vez que se crea alguno de estos métodos, en las revistas de informática suelen dar toda la información necesaria para saber en qué se basa el nuevo sistema anticopia y cómo combatirlo.

e) Cualquier otro sistema de protección siempre que la empresa desarrolladora informe sobre cómo lo ha implementado. Éste es, por ejemplo, el caso de Microsoft que cada vez que protege su software "saca pecho" por ello dando todos los detalles al respecto. El ejemplo más reciente ha sido el del Service Pack para Windows XP, del cual han dicho que incluye como seguridad una lista con todas los números de serie que se han distribuido por Internet. En fin, no hace falta ser un hacha para imaginar como saltarse la seguridad...

Como podemos ver, usando la lógica es muy posible saltar la seguridad de determinados programas de forma más sencilla que empleando la IIS ya que, como veremos en futuros artículos, aunque ésta se basa en ideas y conceptos sencillos, su aplicación se hace bastante pesada y engorrosa. Es por ello que los programadores deberemos de tener en mente la importancia de que la base de la seguridad de nuestro software debe ser desconocida para los demás. De no ser así, estaremos abriendo puertas ocultas a los invasores de nuestras creaciones.

Ingeniería Inversa del Software

- Parte III -

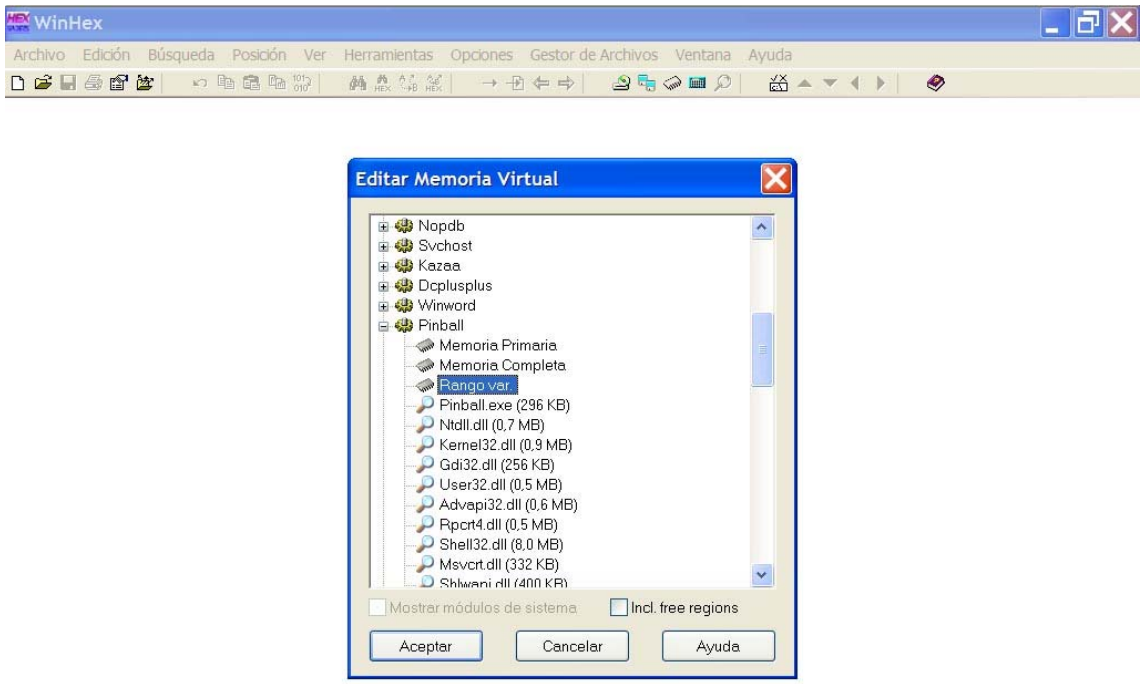
El editor de memoria

Como ya se comentó en el primer escrito de esta colección sobre la IIS, el editor de memoria es una utilidad que permite localizar y modificar el valor de variables que se encuentran en la memoria RAM del sistema. En estas variables se almacenan todos los datos que maneja un programa, incluso la información sobre si está registrado o cuantos días de prueba quedan. De la misma manera, este tipo de editor también se puede emplear para hacer trampas en los juegos, lo cual no deja de ser un ataque al software.

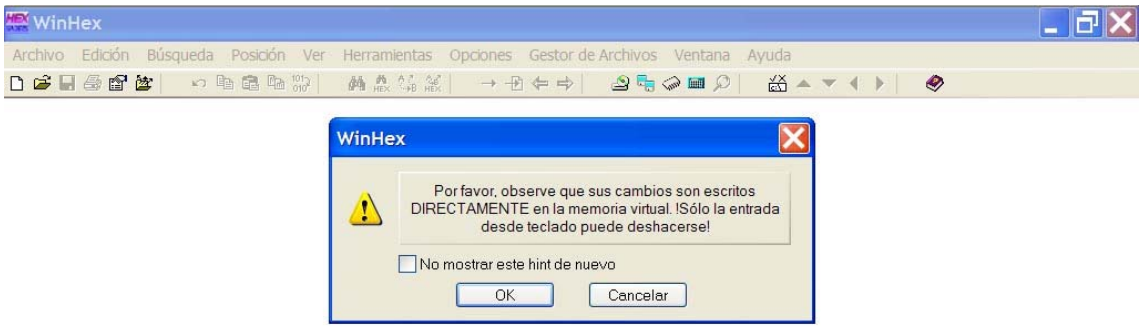
Cuando se trabaja directamente con la memoria del sistema no se dispone de la lista de nombres de las variables ni nada por el estilo, solo se tienen códigos hexadecimales representando sus valores. Por lo tanto, la parte más compleja del proceso que se llevará a cabo para modificar el valor de una variable de un programa consiste en conocer la posición donde está almacenada en memoria esta variable.

En este reportaje se pretendía mostrar como se puede modificar una variable (la puntuación) de un programa (el Pinball de Windows) con un editor de memoria (WinHex), pero debido a la incompatibilidad con Windows XP de los editores de memoria que conozco y manejo (WinHex y GameWizard32) no me es posible mostrar el desarrollo completo, por lo que se intentará clarificar al máximo las partes que no se han podido realizar. Los pasos a seguir en este proceso son las siguientes:

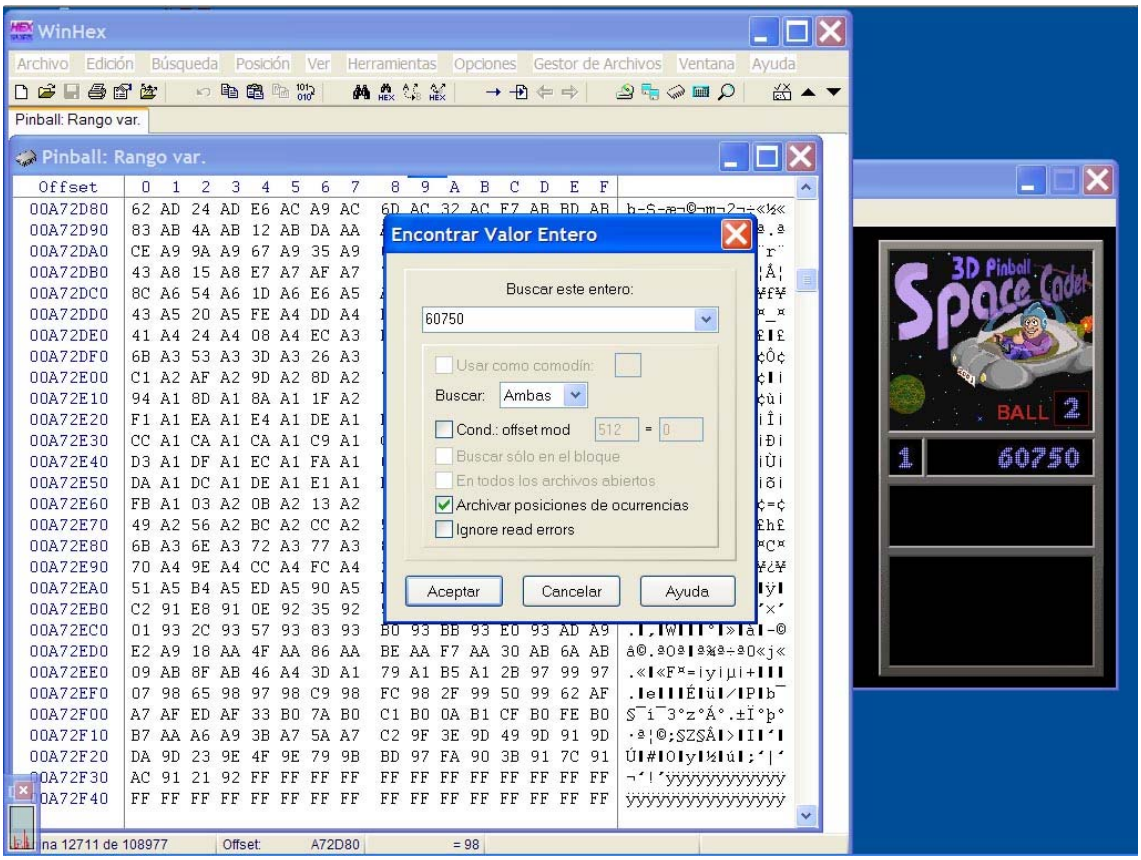
1. Primero hay que seleccionar sobre qué proceso en ejecución se trabajará.



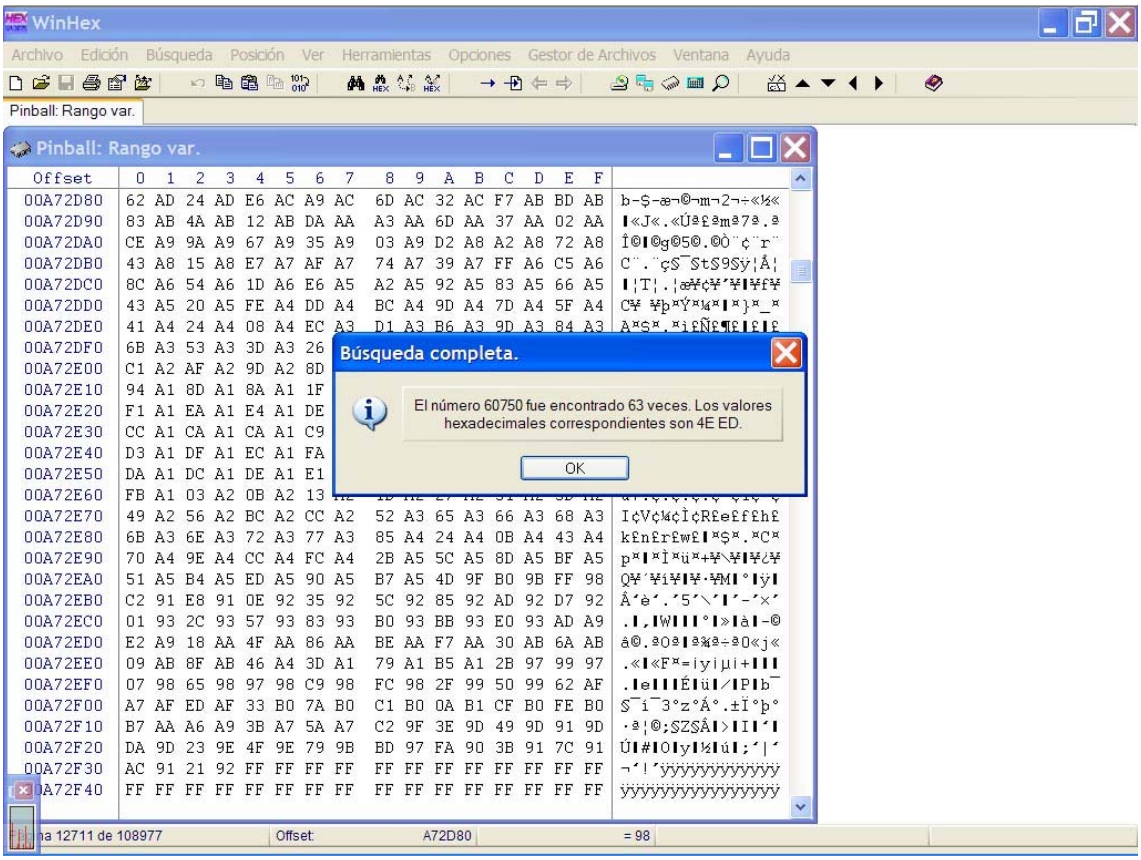
Nota: el propio editor avisa del riesgo que implica su uso.



2. Se pide al editor que monitorice todas las posiciones de la memoria donde se encuentre el valor que tiene actualmente la variable. Si se trata de un número, hay que indicar en que formato se quiere buscar (byte, short, integer, float, double...) ya que según el formato empleado se almacenará con un código hexadecimal u otro. Éste es el único detalle que puede complicar un poco este acto vandálico, aunque no mucho porque la mayoría de los programadores suelen usar o bien el menor tamaño de dato que les valga, o bien el tipo integer siempre que no se necesite otro mayor.



Lo normal es que aparezcan varias ocurrencias de la búsqueda, ya que puede repetirse por casualidad o por copia de valores en otras variables del programa, como por ejemplo, en este caso, en rutinas que manejen la puntuación.



Si el WinHex funcionara bien, aparte de decir el número de ocurrencias debería crear un documento de texto indicando ordenadamente cada una de las direcciones donde se ha encontrado. Esto no ocurre así, por lo que a partir de aquí no podré mostrar más capturas.

3. Al haber localizado más de una aparición del dato buscado, se hace necesario cambiar su valor. En este caso, por ejemplo, se aumenta la puntuación jugando un poco más.

4. Seguidamente, si se usa el WinHex, hay que buscar el nuevo valor de la variable que se quiere controlar y comparar a ojo los dos documentos generados con las listas de direcciones donde ha aparecido.

   Si se usara el GameWizard32, los puntos anteriores habrían sido idénticos, pero en este solo se le tendría que preguntar en que posiciones de las encontradas anteriormente se ha producido un cambio al nuevo valor. Es por tanto un proceso más rápido y sencillo que si se usa el WinHex.

Así habría que repetir estos dos últimos pasos hasta que solo quedara una única posición posible. Si no quedara ninguna posición, está claro que hay que cambiar el tipo de dato y probar con otro.

5. Finalmente, una vez localizada la posición, se va a ella y se modifica manualmente. También es posible crear un script en cualquiera de estos programas para hacer que cuando la variable monitorizada cambie a un determinado valor, se modifique tomando otro valor distinto. Esto es útil, por ejemplo, para conseguir vidas infinitas en un juego.

Jöx€@n Nieto

Ingeniería Inversa del Software

- Parte III -

Modificación de textos en programas (Editores hexadecimales y de recursos)

Tradicionalmente, cuando se ha pretendido modificar un texto que se pueda ver en un programa compilado, se ha utilizado un editor hexadecimal. Últimamente, no obstante, se han puesto muy de moda los editores de recursos que hacen mucho más sencillo e intuitivo este acto. No obstante, los editores hexadecimales siguen siendo mucho más eficaces y no son tan complicados como pueda parecer a simple vista.

El editor hexadecimal es un tipo especial de editor que puede mostrar y modificar cualquier tipo de carácter, incluso los especiales. Esto lo consigue manejando los códigos hexadecimales en vez de los ASCII, aunque estos también son mostrados cuando no son especiales en otra ventana aparte que sirve de referencia. Gracias a esta característica se puede editar cualquier fichero, incluso ejecutables, sin perder ningún carácter.

El editor de recursos es otro tipo de editor que solo muestra las cadenas de texto estáticas que se han definido en tiempo de programación, ocultando de esta manera el resto de caracteres hexadecimales.

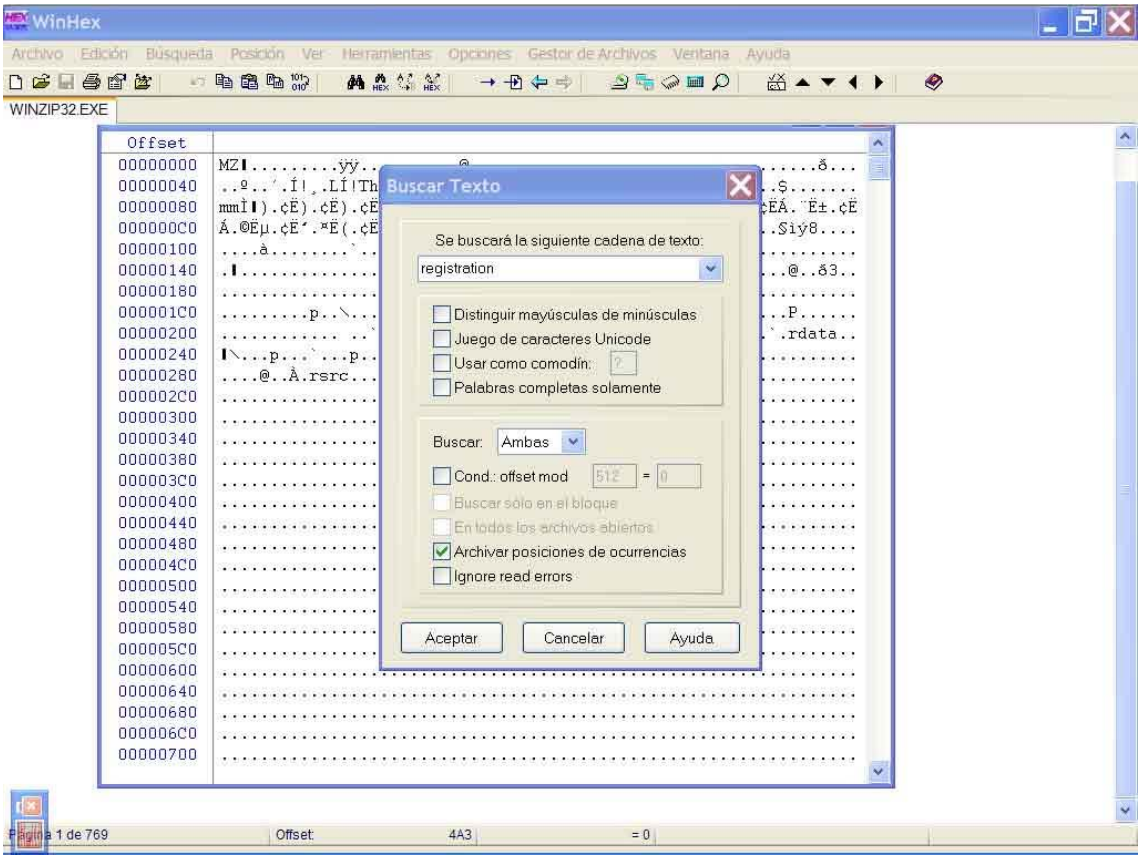
El problema de los editores de recursos es que muchas veces son incapaces de editar un programa que podría editarse hexadecimalmente. Esto es habitual cuando el programa contiene pantallas con caracteres en algún lenguaje que no está instalado en el ordenador, como por ejemplo el chino, japonés, árabe, etc.

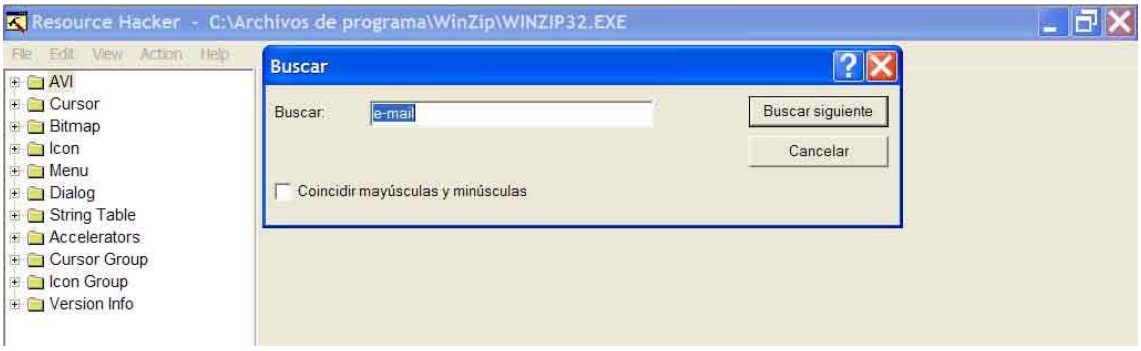
La barrera que no puede saltarse ninguno de estos programas (de momento) es la encriptación. Por tanto, nosotros como programadores que sabemos esto, deberemos encriptar siempre la información importante contenida en nuestro software. Como vamos a ver a continuación, es muy fácil cambiar el nombre del creador del programa, su dirección de email o cualquier otro dato que se pueda proporcionar para cobrar por el uso del software completo.

Veamos como podría hacerse esto con un editor hexadecimal como el WinHex y con un editor de recursos como el Resource Hacker:

1. Se abre con el editor el programa (.exe) o la librería (.dll) correspondiente.

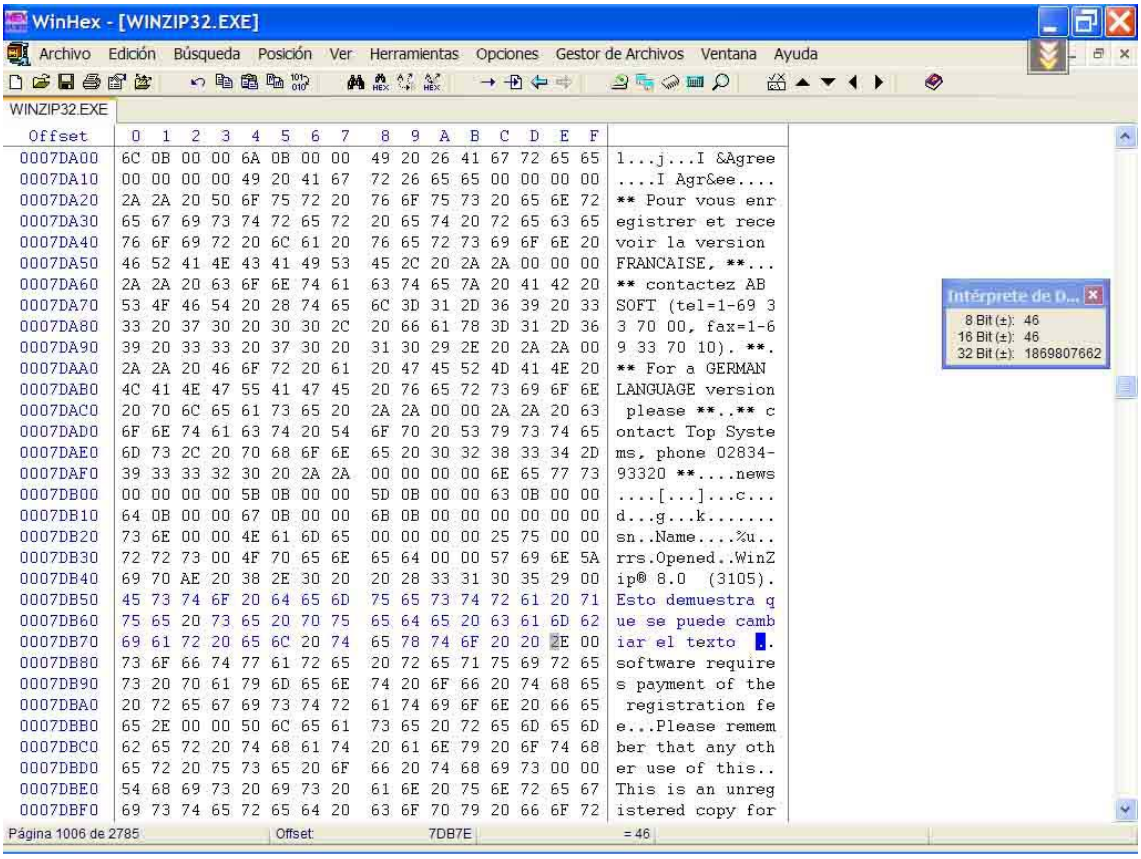
2. Se busca el texto que se quiera modificar.

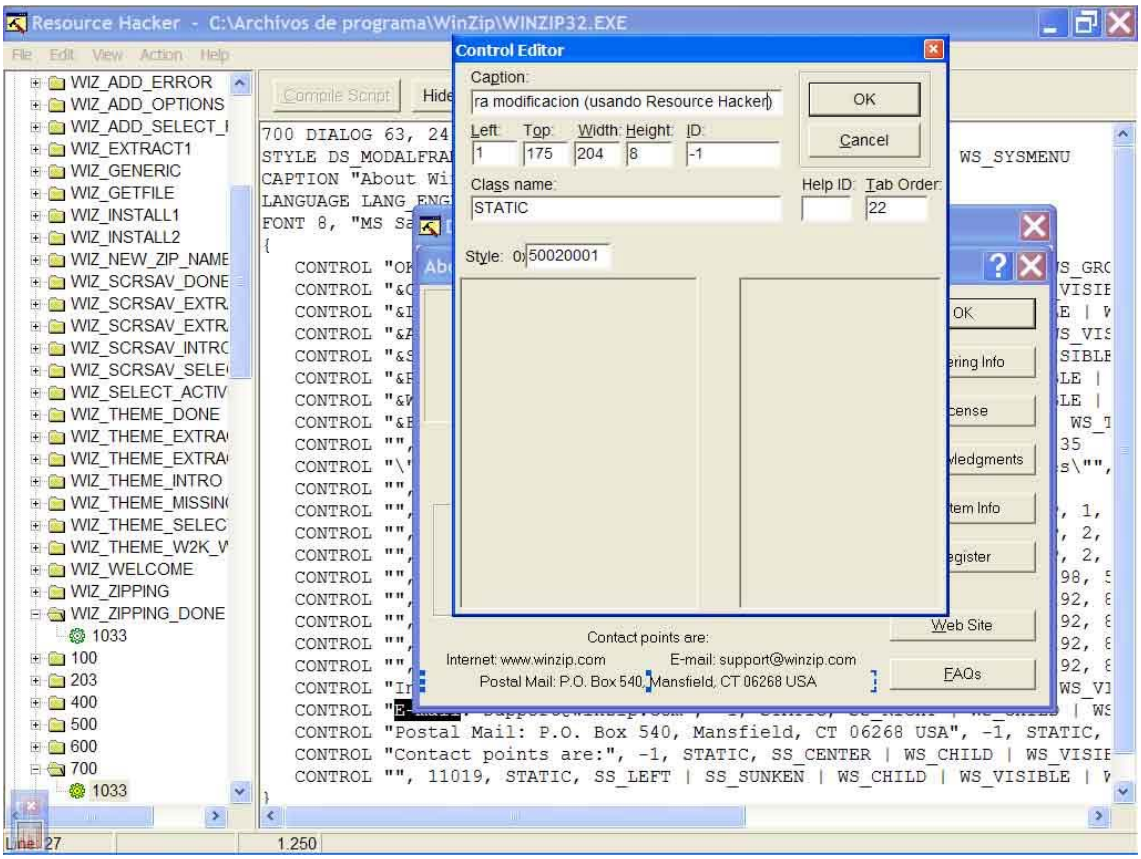




3. Modificar el texto deseado, respetando siempre la longitud de la cadena de texto original si se hace con editor hexadecimal. Se puede escribir un texto más corto y rellenar con espacios, pero no se puede sobrepasar el texto original porque estaríamos sobrescribiendo el código del programa y dejaría de funcionar correctamente. Como los editores de recursos regeneran el

código para acomodar las nuevas cadenas, esta consideración no es necesaria si se usa uno de esos programas.





4. Salvar el archivo modificado.

Al ejecutar el programa se puede observar se puede observar el resultado:

A la vista está que no es nada complicado conseguir "retocar" el texto de un programa si éste no está oculto mediante encriptación, por lo que ésta debe de ser una de nuestras herramientas esenciales de programación. No cuesta nada encriptar los textos importantes de la aplicación y nos protege eficazmente contra cualquier ataque de este tipo.

Jöx€@n Nieto